Delft University of Technology

Fluid Structure Interaction Assignment 1

Delft University of Technology, Delft, South Holland, 2628 CD

Changkyu Park 4646061
Malte Wegener 4672194

Submission Date: March 3, 2021
Course: AE4117

Contents

1. Conservative and consistent coupling approach

1.1. Exercise 1.1

The coupling of fluid and structure equations at the interface is given by

$$\begin{aligned} \mathbf{u}_f &= \mathbf{u}_s \quad \text{on} \quad \Gamma, \\ p_s \mathbf{n}_s &= p_f \mathbf{n}_f \quad \text{on} \quad \Gamma, \end{aligned}$$

where $\mathbf{u}_{f,s}$ is the displacement of this interface, $p_{f,s}$ is the pressure tensor, $\mathbf{n_{f,s}}$ is the outward normal of flow and structure interface and Γ is represents the continuous interface between the fluid and structure. The discrete form is given by

$$\mathbf{U}_f = H_{sf}\,\mathbf{U}_s \quad \text{and} \tag{1.1}$$

$$\mathbf{P}_s = H_{fs}\mathbf{P}_f\,, \tag{1.2}$$

where H_{sf} and H_{fs} are the transformation matrices to translate between interfaces of fluid and structure. As for $\mathbf{U}$ and $\mathbf{P}$, they feature the displacements and pressure of the discretized interface respectively and are expressed as

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n^u} N^i(\mathbf{x})\mathbf{U}_i, \quad p(\mathbf{x})\mathbf{n}(\mathbf{x}) = \sum_{j=1}^{n^p} D^j(\mathbf{x})\mathbf{P}_j, \tag{1.3}$$

where $n_{u,p}$ represents the number of unknowns on the interface for displacement u and pressure p. $N(\mathbf{x})$ represents a function which depends on the method of spatial discretization that is used for the displacement u and $D(\mathbf{x})$ represents a function which depends on the method of discretization that is used for the pressure p.

For the conservative approach, the energy is conserved globally over the interface between fluid mesh f and structure mesh s. The conservation is expressed as

$$\int_{\Gamma_f} \mathbf{u}_f \cdot p_f \mathbf{n}_f \, ds = \int_{\Gamma_s} \mathbf{u}_s \cdot p_s \mathbf{n}_s ds, \tag{1.4}$$

where $\mathbf{u} = \mathbf{x}^n - \mathbf{x}^{n-1}$.

Substituting (1.3) into left hand side of (1.4), we obtain

$$\begin{aligned} \int_{\Gamma_f} \mathbf{u}_f \cdot p_f \mathbf{n}_f \, ds &= \int_{\Gamma_f} \left[\sum_{j=1}^{n_f^u} N_f^j \mathbf{U}_{f_j} \right] \left[\sum_{i=1}^{n_f^p} D_f^i \mathbf{P}_{f_i} \right] ds \\ &= \sum_{i=1}^{n_f^p} \left[\sum_{j=1}^{n_f^u} \int_{\Gamma_f} D_f^i N_f^j \, ds \, \mathbf{U}_{f_j} \right] \mathbf{P}_{f_i} \\ &= \left[M_{ff} \mathbf{U}_f \right]^T \mathbf{P}_f\,, \end{aligned} \tag{1.5}$$

where M_{ff} is defined as

$$M_{ff}^{ij} = \int_{\Gamma_f} D_f^i N_f^j \, ds.$$

Similarly for the right hand side of (1.4), we have

$$\begin{aligned} \int_{\Gamma_s} \mathbf{u}_s \cdot p_s \mathbf{n}_s ds &= \int_{\Gamma_s} \left[\sum_{j=1}^{n_s^u} N_s^j \mathbf{U}_{s_j} \right] \left[\sum_{i=1}^{n_s^p} D_s^i \mathbf{P}_{s_i} \right] ds \\ &= \sum_{i=1}^{n_s^p} \left[\sum_{j=1}^{n_s^u} \int_{\Gamma_s} D_s^i N_s^j \, ds \, \mathbf{U}_{s_j} \right] \mathbf{P}_{s_i} \\ &= \left[M_{ss} \mathbf{U}_s \right]^T \mathbf{P}_s, \end{aligned} \tag{1.6}$$

where M_{ss} is defined as

$$M_{ss}^{ij} = \int_{\Gamma_s} D_s^i N_s^j ds$$

Putting (1.4), (1.5) and (1.6) together, we obtain

$$\left[M_{ff}\mathbf{U}_f\right]^T \mathbf{P}_f = \left[M_{ss}\mathbf{U}_s\right]^T \mathbf{P}_s. \tag{1.7}$$

Substituting (1.1) into the left hand side of (1.7), we get

$$\begin{aligned}
\left[M_{ff}H_{sf}\mathbf{U}_s\right]^T \mathbf{P}_f &= \left[M_{ss}\mathbf{U}_s\right]^T \mathbf{P}_s \\
\mathbf{U}_s^T H_{sf}^T M_{ff}^T \mathbf{P}_f &= \mathbf{U}_s^T M_{ss}^T \mathbf{P}_s \\
H_{sf}^T M_{ff}^T \mathbf{P}_f &= M_{ss}^T \mathbf{P}_s \\
\left(M_{ss}^T\right)^{-1} H_{sf}^T M_{ff}^T \mathbf{P}_f &= \mathbf{P}_s.
\end{aligned} \tag{1.8}$$

Since $\left(A^{-1}\right)^T = \left(A^T\right)^{-1}$ holds for an arbitrary matrix A, (1.8) can be further expressed as

$$\left(M_{ff}H_{sf}M_{ss}^{-1}\right)^T \mathbf{P}_f = \mathbf{P}_s. \tag{1.9}$$

Comparing (1.2) to (1.9), it can be deduced and proved that

$$H_{fs} = \left(M_{ff}H_{sf}M_{ss}^{-1}\right)^T. \tag{1.10}$$

2. Analytic problem

2.1. Exercise 2.1

In setting up the matrices H_{sf} and H_{fs} for both the Nearest Neighbour (NN) and Radial basis function interpolation (RBF) methods, the following lines of codes were written. The skeleton of the program was adapted from the provided *Analyt.m* program.

```
%% Nearest Neighbor Interpolation structure -> flow
HsfNN = zeros(Nf, Ns);
for i=1:Nf
    dist = sqrt((Xs(:,1)-Xf(i,1)).^2 + (Xs(:,2)-Xf(i,2)).^2); % Calculate dist of i-th fluid point
        to every structure point
    [Min, Index] = min(dist); % Obtain the index of the smallest distance
    j = Index; % set j as the index
    HsfNN(i,j) = 1;
end

%% Nearest Neighbor Interpolation flow -> structure
HfsNN = zeros(Ns, Nf);
for i=1:Ns
    dist = sqrt((Xs(i,1)-Xf(:,1)).^2 + (Xs(i,2)-Xf(:,2)).^2); % Calculate dist of i-th structure
        point to every fluid point
    [Min, Index] = min(dist); % Obtain the index of the smallest distance
    j = Index; % set j as the index
    HfsNN(i,j) = 1;
end

%% RBF interpolation structure -> flow
PHI = zeros(Ns,Ns);
P = [ones(Ns,1), Xs];
```

```
for i=1:Ns
    for j=1:Ns
        diff = Xs(i,:) - Xs(j,:); % vector of difference in x and y
        Norm = sqrt(diff(1)^2 + diff(2)^2); % 2-Norm
        if Norm <= 1
            PHI(i,j) = (1-Norm)^4*(4*Norm+1);
        end
    end
end
C   = [PHI P ; P' zeros(3)];
PHI = zeros(Nf,Ns);
P = [ones(Nf,1), Xf];
for i=1:Nf
    for j=1:Ns
        diff = Xf(i,:) - Xs(j,:); % vector of difference in x and y
        Norm = sqrt(diff(1)^2 + diff(2)^2); % 2-Norm
        if Norm <= 1
            PHI(i,j) = (1-Norm)^4*(4*Norm+1);
        end
    end
end
HsfRBF = [PHI P]/C;
HsfRBF = HsfRBF(:,1:Ns);

%% RBF interpolation flow -> structure
PHI = zeros(Nf);
P = [ones(Nf,1), Xf];
for i=1:Nf
    for j=1:Nf
        diff = Xf(i,:) - Xf(j,:); % vector of difference in x and y
        Norm = sqrt(diff(1)^2 + diff(2)^2); % 2-Norm
        if Norm <= 1
            PHI(i,j) = (1-Norm)^4*(4*Norm+1);
        end
    end
end
C   = [PHI P ; P' zeros(3)];
PHI = zeros(Ns,Nf);
P = [ones(Ns,1), Xs];
for i=1:Ns
    for j=1:Nf
        diff = Xs(i,:) - Xf(j,:); % vector of difference in x and y
        Norm = sqrt(diff(1)^2 + diff(2)^2); % 2-Norm
        if Norm <= 1
            PHI(i,j) = (1-Norm)^4*(4*Norm+1);
        end
    end
end
HfsRBF = [PHI P]/C;
HfsRBF = HfsRBF(:,1:Nf);
```

Afterwards, the row sums of these 4 matrices were calculated using the following lines of codes

```
%% Compute row sums of transformation matrices
HsfNN_rsum = sum(HsfNN,2);
HsfRBF_rsum = sum(HsfRBF,2);
HfsNN_rsum = sum(HfsNN,2);
HfsRBF_rsum = sum(HfsRBF,2);
```

It was observed that every row of all 4 matrices has a row sum of 1. This then implies that all the matrices are consistent.

2.2. Exercise 2.2

Since energy is conserved for the conservative methods, it is expected that the error in computational work is 0 for the conservative NN and RBF methods. To check this, the L_2-error was computed as follows in *Analyt.m* where *cv* stands for conservative method. Above this, the lines of code for pressure computation has been added to complete the program.

```
%% Compute pressures - consistent approach
Ps_NN = HfsNN * Pf;
Ps_RBF = HfsRBF * Pf;
Pf_ex = sin(2*pi*Xi);

%% Compute error in work at interface (work on fluid side should be equal)
err_dW_NN   = abs( (Mf*Pf)' * Df_NN - (Ms*Ps_NN)' * Ds );
err_dW_NN_cv = abs( (Mf*Pf)' * Df_NN - (Ms*Ps_NN_cv)' * Ds );
err_dW_RBF  = abs( (Mf*Pf)' * Df_RBF - (Ms*Ps_RBF)' * Ds );
err_dW_RBF_cv = abs( (Mf*Pf)' * Df_RBF - (Ms*Ps_RBF_cv)' * Ds );
```

The following are the results

$$\text{err\_dW\_NN} = 1.6109 \cdot 10^{-4}$$
$$\text{err\_dW\_NN\_cv} = 5.5511 \cdot 10^{-17}$$
$$\text{err\_dW\_RBF} = 3.0357 \cdot 10^{-5}$$
$$\text{err\_dW\_RBF\_cv} = 0$$

Since the value of 10^{-17} can be treated as machine zero, both the conservative methods resulted in an error of 0 in computational work as expected.

2.3. Exercise 2.3

The following figures from Figure 1 till Figure 7 are the outcome of the *loop.m* program. Starting with Figure 1, it describes the displacement in the fluid points with respect to x-location for exact solution and solution obtained using Nearest Neighbour (NN) and Radial basis function (RBF) methods. A sinusoidal shape can be observed for all 3 lines.

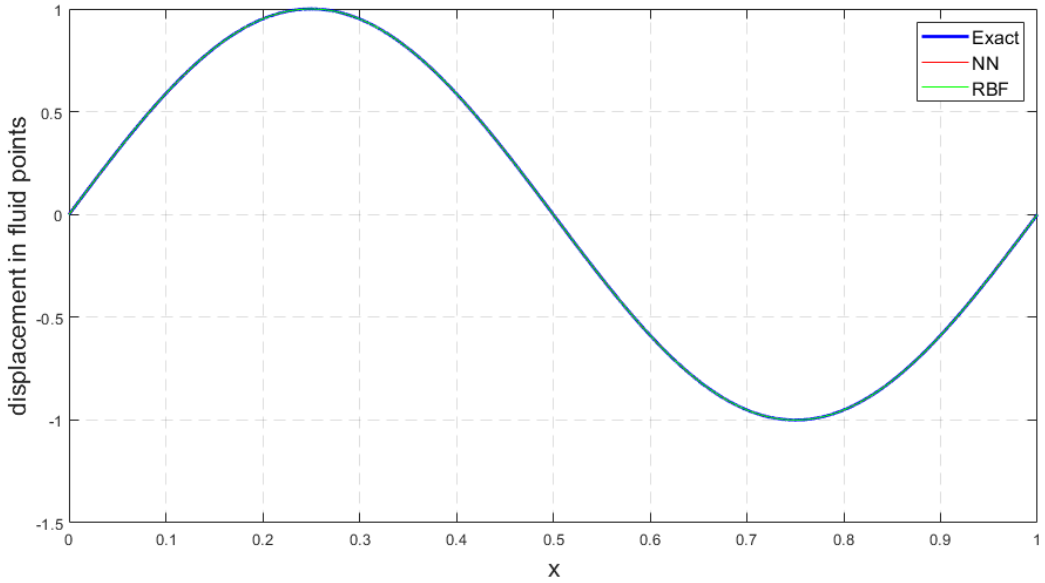

Fig. 1 Displacement in fluid points

This figure was then zoomed in to inspect the curves more closely and it is shown in Figure 2. It is observed that the RBF curve exactly overlaps with the curve of the exact solution whereas the NN curve fluctuates around this exact solution curve.

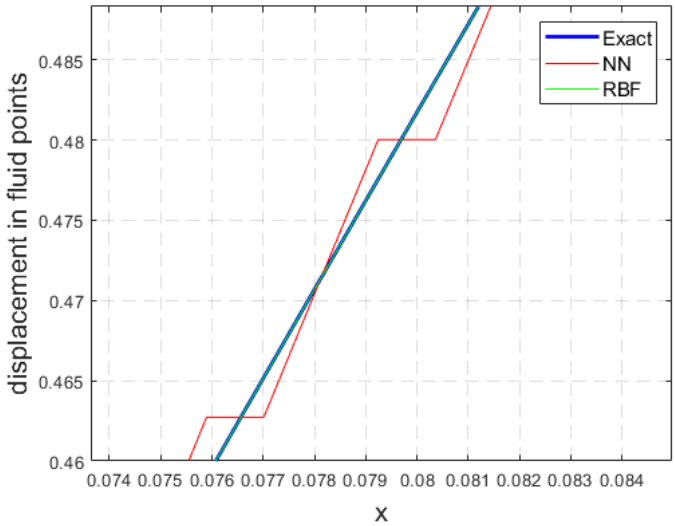

Fig. 2 Figure 1 zoomed in

Figure 3 describes the pressure in fluid points with respect to the x-location for exact solution and solution obtained using conservative and non-conservative NN and RBF methods. Just like Figure 1, this figure also portrays sinusoidal behaviour. Again, a zoomed-in plot has been made to closely investigate the behaviour of the curves in Figure 4.

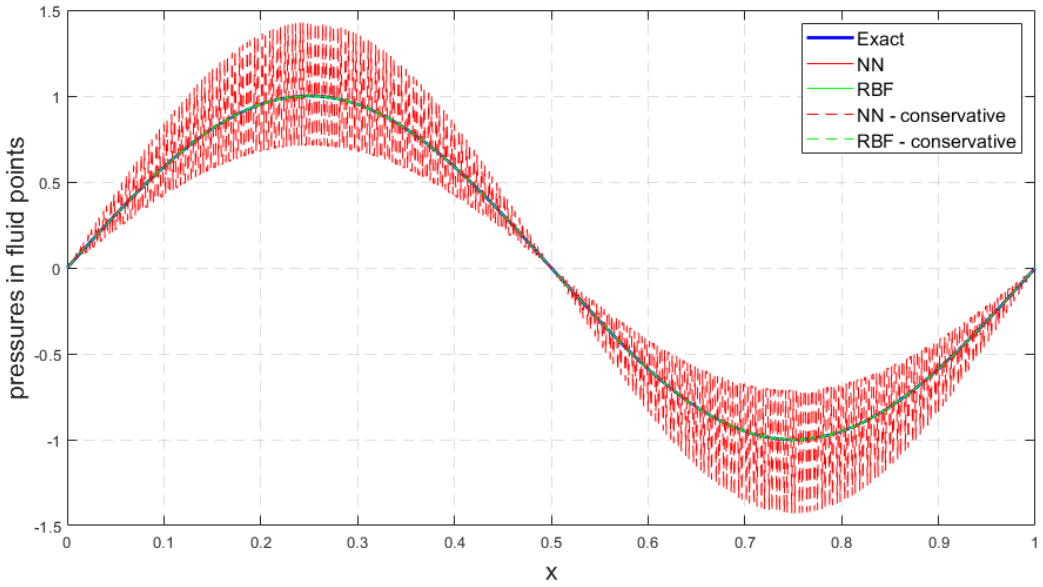

Fig. 3 Pressure in fluid points

It is observed that the conservative NN and RBF methods fluctuates more than their non-conservative counterparts.

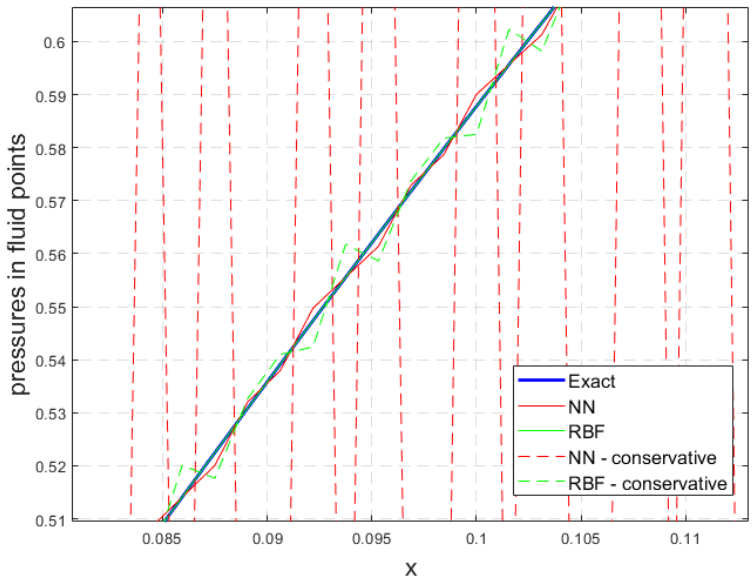

Fig. 4 Figure 3 zoomed in

Moving onto the error plots, Figure 5 shows the displacement L_2-error in fluid points for the NN and RBF methods. The RBF methods show a higher order of accuracy compared to the NN method. It can be seen that NN is first-order accurate while RBF is third-order accurate. Thus, although the RBF methods start at a slightly higher error, as the mesh gets finer, its error diminishes much more quickly than NN and hence ending up at a much lower error for the greatest refinement.

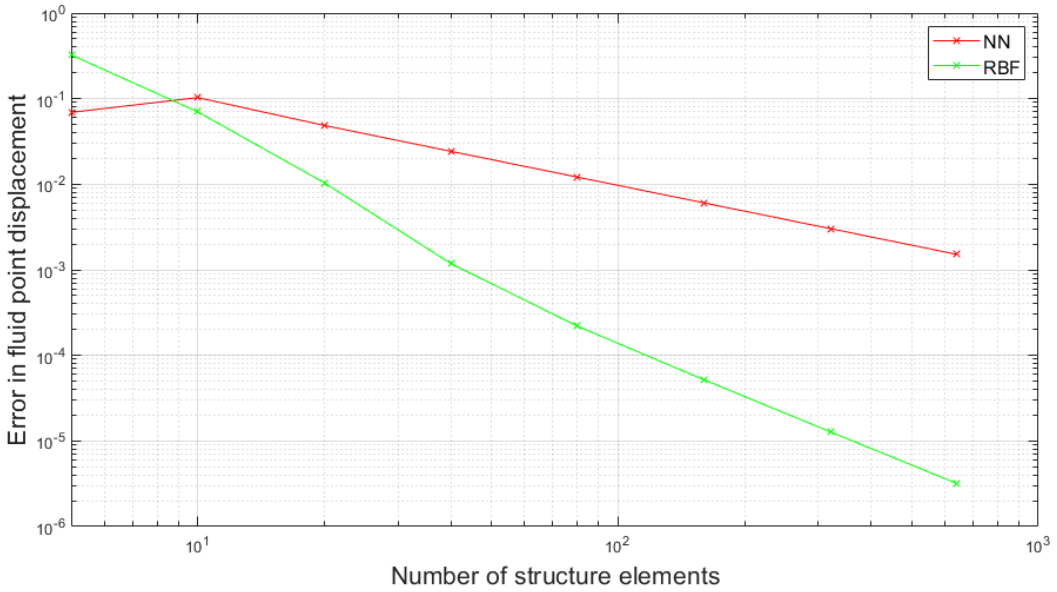

Fig. 5 Displacement error in fluid points

Next, Figure 6 shows the pressure L_2-error in structure points for the conservative and non-conservative NN and RBF methods. As for the non-conservative methods, they show similar behaviour as they did for Figure 5 whereby NN is first-order accurate while RBF is approximately second-order accurate. As for the conservative RBF method, it does show third-order accurate behaviour for the coarse mesh but after about a number of structure elements of $4 \cdot 10^1$, it shows a 0 convergence rate. On the other hand, the conservative NN method shows this 0 convergence rate throughout the entire mesh refinement.

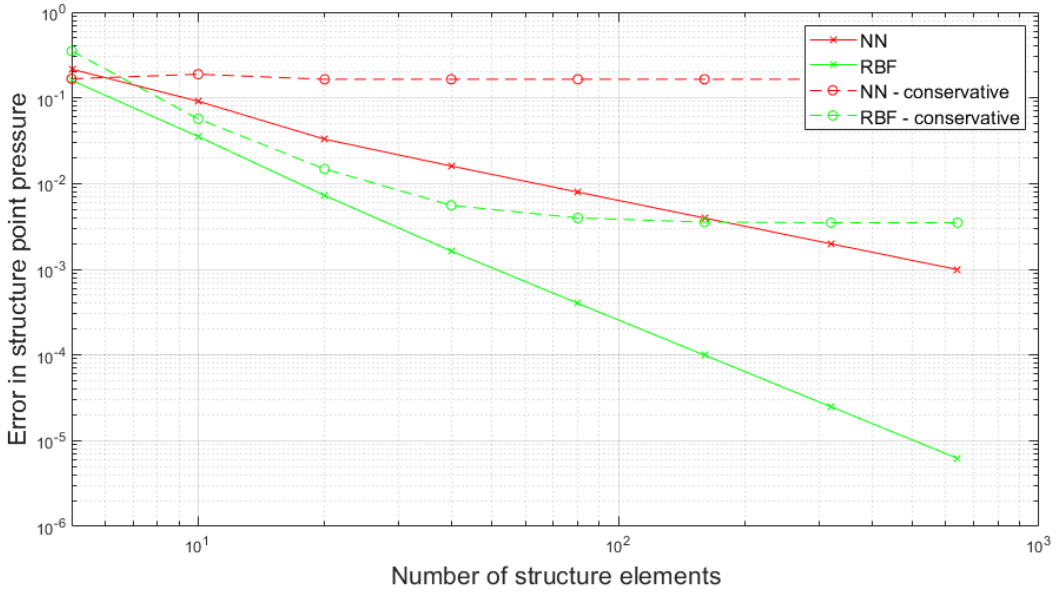

Fig. 6 Pressure error in structure points

Lastly, the following Figure 7 features the L_2-error in work at the interface of the fluid and structure mesh for conservative and non-conservative NN and RBF methods. In Exercise 2.2, it was discussed that an error of 0 is expected

for the work of the conservative method since this method conserves work over the interface. This is clearly represented in the figure whereby the conservative curve lies below the error magnitude of machine precision of approximately 10^{-16}. A few points are missing from these curves as they were found to have an exact 0 value which was included in the plot. Thus, the error for conservative methods came out as expected. As for the non-conservative counterparts, NN has an order of accuracy of about 2.5 and RBF is fourth-order accurate. Moreover, they are not close to the machine precision and this is also expected since the work is not explicitly conserved in the non-conservative method.

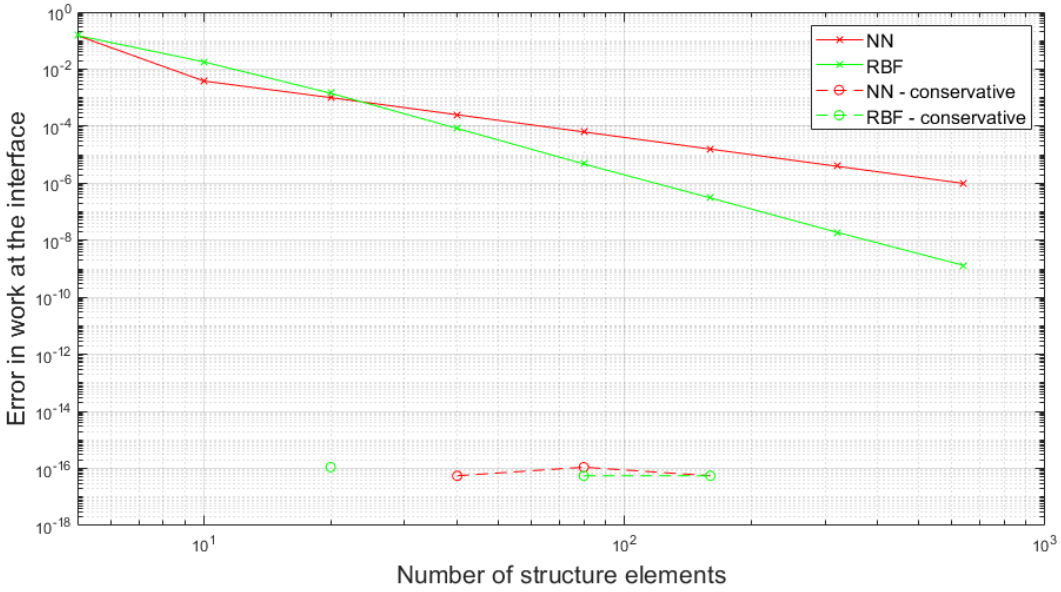

Fig. 7 Work error in the interface

These errors are sourced from interpolation and discretization errors. The interpolation errors exist due to the non-matching meshes of fluid and structure interface while the discretization errors exist due to the coarseness of the mesh. Due to the steep increment in computational work with increasing fineness of the mesh, only a sufficiently large number of elements were taken into account, thus the discretization error was not completely eliminated. Hence, even when matching meshes are selected, this discretization error will still be present although the interpolation error can be completely removed.

3. Mesh motion for two-dimensional mesh

3.1. Exercise 3.1

The displacement interpolation was implemented into the matlab script with a thin plate spline as the radial basis function. The thin plate spline is given by

$$\phi(\|\mathbf{x}\|) = \|\mathbf{x}\|^2 \ln \|\mathbf{x}\| \tag{3.1}$$

The total interpolation function was constructed as a sum of radial basis functions and a polynomial $q(\mathbf{x}) = c_0 + c_x x + c_y y$

$$s(\mathbf{x}) = \sum_{j=1}^{N_b} \gamma_j \phi\left(\left\|\mathbf{x} - \mathbf{x}_{b_j}\right\|\right) + q(\mathbf{x}) \tag{3.2}$$

The displacement was separated into displacements in x and displacements in y and a separate interpolant was created for each. The working of the implementation was checked for the given parameters and the resulting orthogonality plot vs time is shown in Figure 8. The deformed mesh can be seen in Figure 9.

The method was implemented in *movemesh.m*

```
%% IMPLEMENT YOUR RBF MESH DEFORMATION HERE

% Set up the RBF matrix and constraints
% rbfMat = [ Mb P
%            PT 0 ]
%
Mb = zeros(Nb,Nb);
for i=1:size(Mb,1)
    for j=1:size(Mb,2)
        n = norm(Xb(i,:)-Xb(j,:));
        if (n > 0)
            Mb(i,j) = n^2*log(n);
        else
            Mb(i,j) = 0;
        end
        end
    end
end

L = ones(Nb, 3);
for i=1:Nb
    L(i,2) = Xb(i,1);
    L(i,3) = Xb(i,2);
end

rbfMat = [[Mb, L];[transpose(L), zeros(3,3)]];
```

```
% constraints = displacements of the boundary nodes (separate in x
% and y)
%
Dx = zeros(Nb+3,1);
Dy = zeros(Nb+3,1);
for i=1:Nb
    Dx(i) = Db(i,1);
    Dy(i) = Db(i,2);
end
```

```
% evaluate RBF function for each internal node
Mb = zeros(Ni,Nb);
for i=1:size(Mb,1)
    for j=1:size(Mb,2)
        n = norm(Xi(i,:)-Xb(j,:));
        if (n > 0)
            Mb(i,j) = n^2*log(n);
        else
            Mb(i,j) = 0;
        end
    end
end

L = ones(Ni, 3);
for i=1:Ni
    L(i,2) = Xi(i,1);
    L(i,3) = Xi(i,2);
end
```

```
A = [Mb,L];

Di(:,1) = A*Px;
Di(:,2) = A*Py;

% Convert internal displacements to global disp vector
for i=1:Ni
    disp(ID(i),2:3) = Di(i,:);
end
```

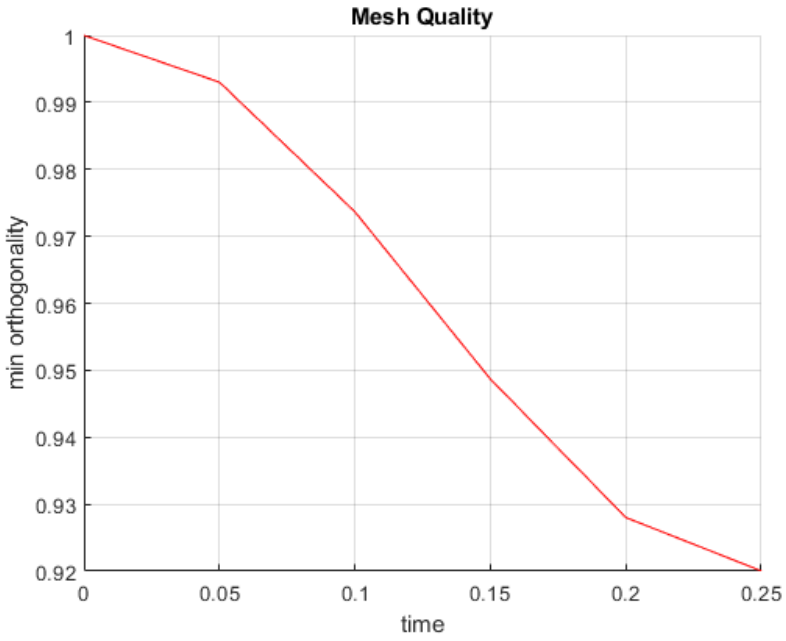

Fig. 8 Mesh orthogonality vs time

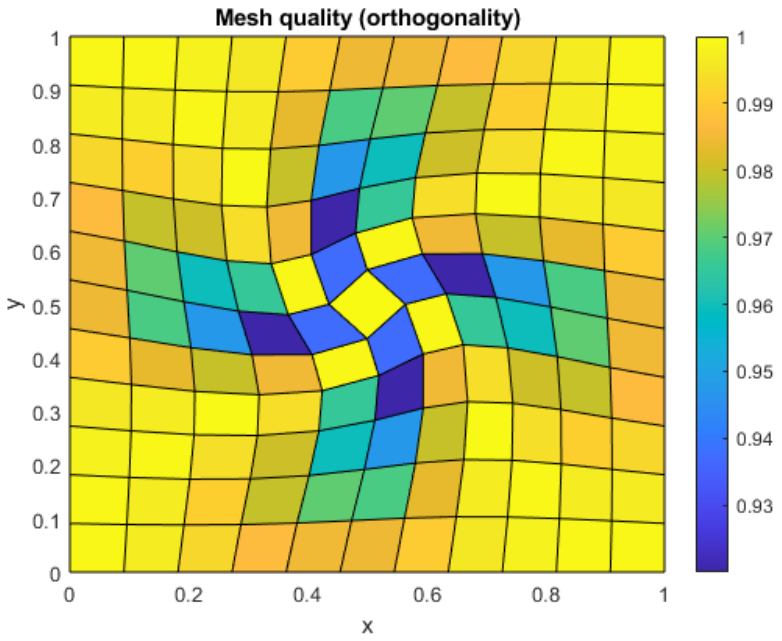

Fig. 9 Mesh orthogonality at maximum rotation

3.2. Exercise 3.2

Absolute and relative displacements were then compared for different numbers of oscillations according to Table 1. The results can be seen in Figure 10.

Table 1 Simulation settings

case	Np	absdisp
1	0.25	1
2	0.25	0
3	2.0	1
4	2.0	0

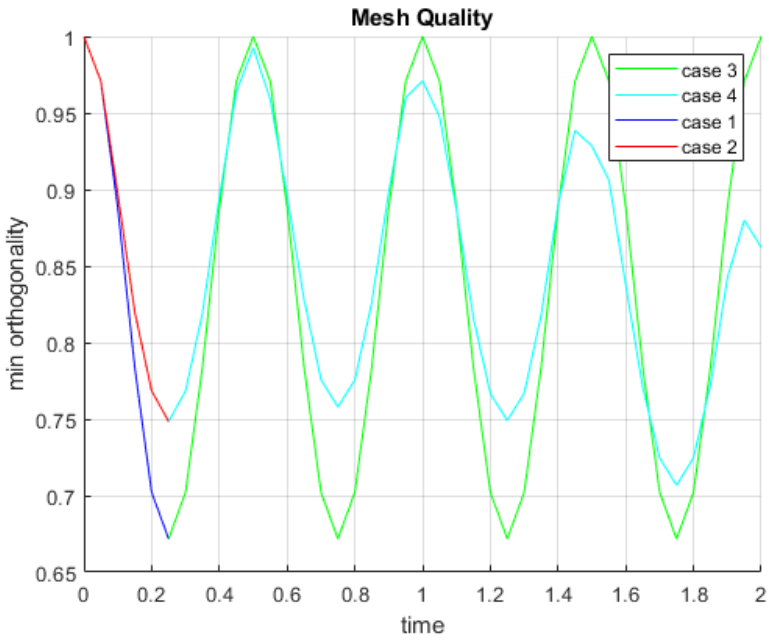

Fig. 10 Mesh orthogonality vs time for different cases

From the figure, it can be seen that for a single movement a relative displacement method performs better than an absolute displacement method. It can be seen that the minimum orthogonality is about 0.08 better for the relative displacement calculation. When considering periodic movements, however, it can be seen that even though the relative displacement method performs better in the beginning, the quality of the mesh rapidly deteriorates for later periods. If the simulation time was to be increased, the quality of the mesh will decrease further. The quality of the mesh will approach 0 when the simulation is kept running for a large number of periods. Thus it can be concluded that for periodic deformations an absolute method is preferred as the mesh quality will be constant for every deformation.

3.3. Exercise 3.3

The influence of timesteps was investigated for different cases given in Table 2 and the results are shown in Figure 11.

Table 2 Simulation settings

case	NdtP	absdisp
1	40	1
2	20	1
3	10	1
4	5	1
5	40	0
6	20	0
7	10	0
8	5	0

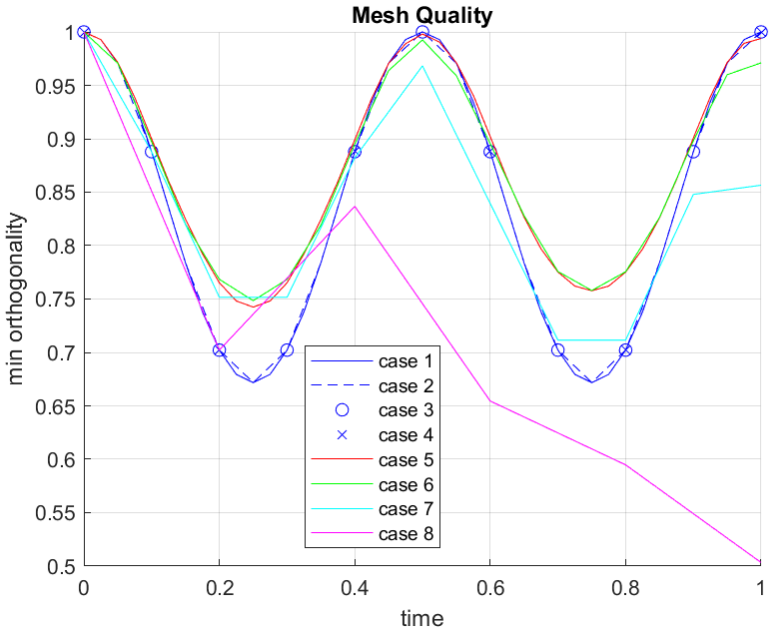

Fig. 11 Mesh orthogonality vs time for different cases

For the absolute displacement, it can be seen that the resulting mesh quality does not depend on the number of steps taken. This is expected, as each displacement is independent of earlier displacements. For the relative displacement, a dependence on the number of time steps can be seen. In the first quarter period all 3 cases, except case 8, perform similarly well. However, when returning to a non-deformed state, it can be seen that the cases with a higher number of time steps return to a better mesh quality and thus perform better in the next deformation. It can also be seen that case 8 performs worse than every other case, as the time step chosen is too big for the frequency of displacement

3.4. Exercise 3.4

From this, it can be concluded that relative displacement methods perform better for single displacements. For this, a time step lower than the time is taken to maximum displacement shall be chosen. When however dealing with periodic displacements absolute displacement methods perform better as these methods will always return to the original mesh when no displacement is present.

3.5. Exercise 3.5

In order to evaluate the performance of compact supported radial basis functions, an interpolant was chosen such that

$$s(\mathbf{x}) = \sum_{j=1}^{N_b} \gamma_j \phi\left(\left\|\mathbf{x} - \mathbf{x}_{b_j}\right\|\right) \tag{3.3}$$

$$\phi(\|\mathbf{x}\|) = \begin{cases} (1 - \|\mathbf{x}\|/r)^4(4\|\mathbf{x}\|/r + 1) & : \|\mathbf{x}\| \leq r \\ 0 & : \|\mathbf{x}\| > r \end{cases} \tag{3.4}$$

. A variable to switch the interpolation method was added into *movemesh.m*

```
if (tps)
    if (n > 0)
        Mb(i,j) = n^2*log(n);
    else
        Mb(i,j) = 0;
    end
```

```
else
    if (n<=r)
        Mb(i,j) = (1-n/r)^4*(4*n/r+1);
    else
        Mb(i,j) = 0;
    end
end
```

```
if(tps)
    rbfMat = [[Mb, L];[transpose(L), zeros(3,3)]];
else
    rbfMat = Mb;
end
```

```
if(tps)
    Dx = zeros(Nb+3,1);
    Dy = zeros(Nb+3,1);
    for i=1:Nb
        Dx(i) = Db(i,1);
        Dy(i) = Db(i,2);
    end
else
    Dx = zeros(Nb,1);
    Dy = zeros(Nb,1);
    for i=1:Nb
        Dx(i) = Db(i,1);
        Dy(i) = Db(i,2);
    end
end
```

The mesh quality was then varied for a single period with NdtP = 20 and a maximum rotation of 90 degrees as shown in Table 3. The results can be found in Figure 12.

Table 3 Simulation settings

case	r	absdisp
1	0.15	1
2	0.25	1
3	0.5	1
4	1	1
5	0.15	0
6	0.25	0
7	0.5	0
8	1	0

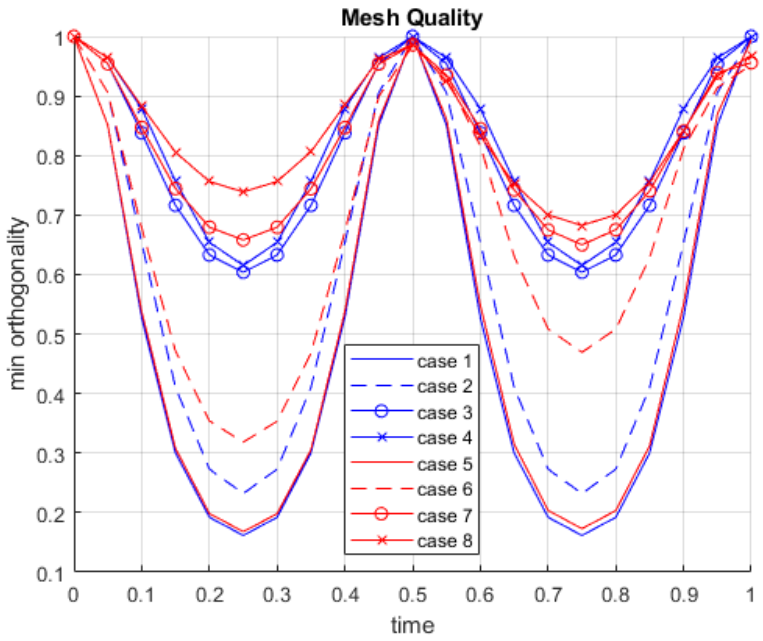

Fig. 12 Mesh orthogonality vs time for different cases

From the figures it can be seen that for a support radius of 0.15 or 0.25, the mesh quality is very low, leading to unusable meshes. For support radii of 0.5 or higher, the quality increases but is still lower than for the previously discussed thin-plate spline interpolation. In the comparison between absolute and relative displacement methods, similar trends as for previously discussed methods can be seen. It can however be seen, that during the second displacement case 6 performs better than in a previous case, this can be explained by the fact that during multiple displacements, mesh elements move to the centre and thus enter the support radius of the basis function increasing mesh quality. In conclusion, this method does not provide any benefit in terms of mesh quality for the case at hand.